TECHNICAL SYSTEM INFORMATION

TECH. INFO.

MEMORY REQUIREMENTS

TRSDOS-16 loads into memory at startup and occupies approximately 48K of memory.

At this time TRSDOS-16 also processes your configuration command file (see **APPENDIX B** for information on the Configuration Command File).

The configuration file tells TRSDOS-16 what files to load into memory for your use. This might include RUNCOBOL, and the Debugger. TRSDOS-16 loads these files into the memory area reserved for TRSDOS-16.

Any memory following the configured files is available to you, the user.

Note: The Editor, Assembler, and Linker load into user memory, not the Configurator memory.

RELATIVE ADDRESSING

TRSDOS-16 loads into memory in a way that is "invisible" to the user. This is TRSDOS-16's way of protecting itself from being modified by other programs.

TRSDOS-16 internally keeps track of two addresses:

- the BASE address - this is the bottom of user memory. (Memory below this address is reserved for TRSDOS-16 and files designated by the configured file.
- the BOUNDS address - this is the top address of memory. (H'1FFFF in a 128K Model 16, H'3FFFF in a 256K Model 16, H'5FFFF in a 384K Model 16, and H'7FFFF in a 512K Model 16.)

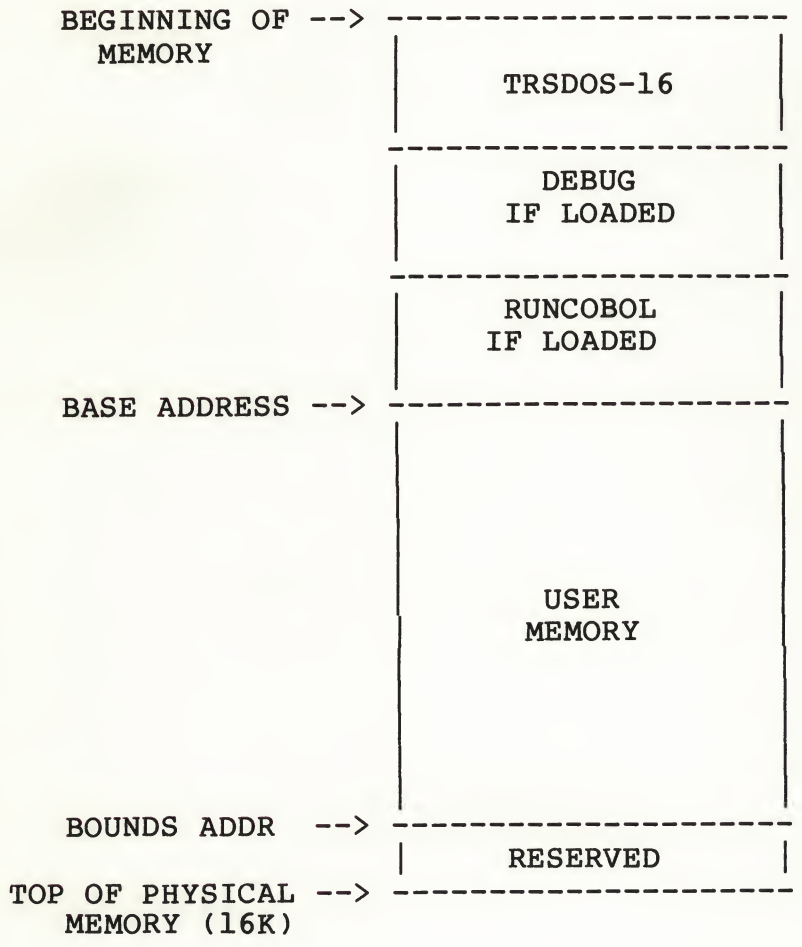
In short:

BOUNDS Address - BASE Address = User Memory

TRSDOS-16 uses the BASE Address to find the physical loading address of your program. When you write a machine-language program, you assign it a "relative loading" address, usually zero. TRSDOS-16 then adds the BASE address to this relative address to determine the physical address where it will load your program:

BASE Address + PROGRAM RELATIVE ADDRESS = PHYSICAL LOAD ADDRESS

TRSDOS-16 uses the BOUNDS address to find the amount of available user memory. This is how the Operating System determines if your program will fall out of the "bounds" of user memory.



MEMORY CHART

DISK ORGANIZATION

FLOPPY DISKETTE

TRSDOS-16 can use either single-sided (double density) or double-sided (double density) diskettes. However, if you are using an Enhanced Model II, you can use only single-sided diskettes because of the type of drives you have.

TRSDOS-16 will automatically format either single or double-sided diskettes according to the type of diskette.

Each side of the double-sided diskette contains 77 tracks, numbered Ø - 76. Therefore, the double-sided diskette can be thought of as one diskette with 154 tracks.

The single-sided diskette has 77 tracks on one side only.

Each track is made up of 32 sectors, numbered 1-32. Each sector contains 256 bytes, except for track Ø (one side only on the double-sided diskette). This track Ø is reserved for System use and is formatted single-density. It contains 26 sectors and 128 bytes per sector.

The total capacity of a double-sided diskette is:

(153 * 32 * 256) + (1 * 26 * 128) = 1,256,7Ø4 bytes

The total capacity of a single-sided diskette is:

(76 * 32 * 256) + (1 * 26 * 128) = 625,92Ø

HARD DISK

The TRS-8Ø Hard Disk Drive is organized into 256 "cylinders". Each cylinder is made up of four tracks that have the same radius on each of the four surfaces (4 * 256= 1Ø24 total tracks per hard disk).

Each track contains 34 sectors, numbered 1-34. Each sector contains 256 bytes.

The total capacity of a hard disk is:

(1Ø24 * 34 * 256) = 8,912,896 bytes

DISK SPACE AVAILABLE TO USER

TRSDOS-16 occupies approximately 31 tracks on the floppy diskette and approximately 32 tracks on the hard disk.

Track Ø is reserved by the System on both the floppy and hard disk. The hard disk also reserves track 1 for diagnostics.

The directory and alternate directory on both the floppy diskettes and hard disk reserve a certain number of tracks, determined by the size of the directories (the size is set by the FORMAT command).

To determine the number of tracks these directories will consume, use this formula:

Directory Space Formula	Description
# of filenames / 4 (round result up to next integer value)	to determine number of sectors used by the primary directory
# of sectors / 32 (floppy) # of sectors / 34 (hard) (round result up to next integer value)	to determine number of tracks used by the primary directory
# of tracks * 2	to determine the total number of tracks used for the primary and alternate directories (the alternate directory is simply a copy of the primary directory)

Note: TRSDOS-16 stores 4 filenames per sector in the directory.

For example, if you use this Format command:

```
FORMAT :1 {PW=PASSWORD,SIZE=25Ø}
```

the directories will consume a total of four tracks:

25Ø filenames / 4	=	62.5
round up	=	63
63 sectors / 32 (floppy)	=	1.96
round up	=	2 tracks for each directory
2 * 2	=	4 total tracks used for the primary and alternate directories.

UNIT OF ALLOCATION

All allocation of disk space is made by single sectors. This means that the smallest non-empty TRSDOS-16 file will consist of one sector.

Single-Sided

Diskette	Tracks	Sectors	Bytes
1	76	2,432	622,592
---	1	32	8,192
---	---	1	256

Double-Sided

Diskette	Cylinders	Tracks	Sectors	Bytes
1	76	154	4,896	1,253,376
---	1	2	64	16,384
---	---	1	32	8,192
---	---	---	1	256

Note: Track Ø on the floppy diskette (only one side for double-sided diskettes) is reserved for System use and is not available for user storage. It is formatted single density with 26 sectors that contain 128 bytes each.

Hard Disk	Cylinders	Tracks	Sectors	Bytes
1	256	1Ø24	34,816	8,912,896
---	1	4	136	34,816
---	---	1	34	8,7Ø4
---	---	---	1	256

DISK FILES

METHODS OF FILE ALLOCATION

TRSDOS-16 provides two ways to allocate disk space for files: Dynamic Allocation and Pre-Allocation.

DYNAMIC ALLOCATION

With dynamic allocation, TRSDOS-16 allocates sectors only at the time of write. For example, when a file is first opened for output, no space is allocated. The first allocation of space is done at the first write. Additional space is added as required by subsequent writes.

With dynamically allocated files, unused sectors are de-allocated (recovered) when the file is closed.

Dynamic allocation is the method TRSDOS-16 uses, unless you execute the CREATE system command.

PRE-ALLOCATION

With pre-allocation, the file is allocated a specified number of sectors when it is created. Pre-allocated files can only be created by the system command CREATE.

TRSDOS-16 will dynamically extend (enlarge) a pre-allocated file as needed.

However, TRSDOS-16 will not de-allocate unused sectors when a pre-allocated file is closed. To reduce the size of a pre-allocated file you must copy it to a dynamically allocated file. The COPY system command does this automatically when the destination is a dynamically-allocated file.

RECORD LENGTH

TRSDOS-16 transfers data to and from disks one sector at a time; i.e., in 256-byte blocks. These are the System's "physical" records.

User records or "logical" records are the buffers of data you wish to transfer to or from a file. These can be from 1 to 256 bytes long.

The Operating System will automatically "block" your logical records into physical records which will be transferred to disk, and "de-block" the physical records into logical records which are used by your program.

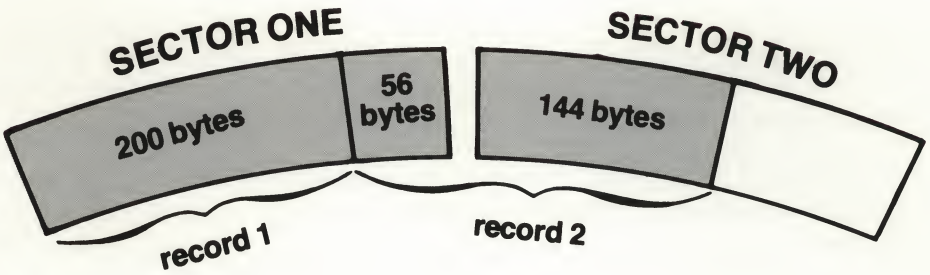
Therefore, your **only** concern during file-access is with logical records. You never need to worry about physical records, sectors, tracks, etc. This is to your benefit, since physical record lengths and features may change in later TRSDOS versions or with other peripheral devices, while the concept of logical records will not.

From this point on, the term "record" refers to a "logical record".

SPANNING

If the record length is not an even divisor of 256, the records will automatically be spanned across sectors.

For example, if the record length is 2ØØ, Sectors 1 and 2 will look like this:



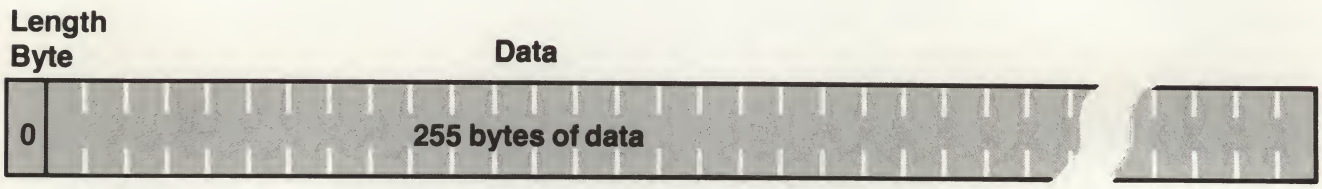
FIXED-LENGTH AND VARIABLE LENGTH RECORDS

TRSDOS-16 files can have either fixed-length or variable-length records. Files with fixed-length records will be referred to as FLRs; files with variable length records, VLRs.

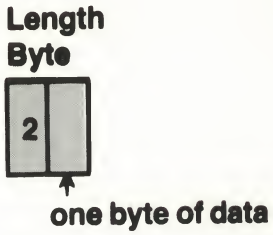
Record length in an FLR file is set when the file is opened for the first time. This length can be any value from 1 to 256 bytes. Once set, the record length in an FLR cannot be changed unless the file is being over-written with new data.

The record-lengths in a VLR file can vary. For example, the first record in a file might have a length of 32; the second, 17; the third, 25Ø; etc.

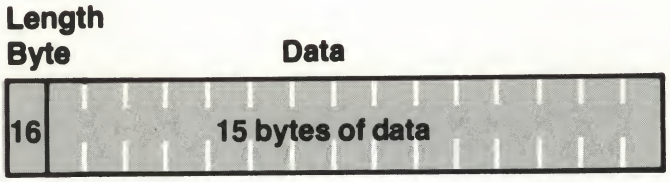
The record length in a VLR file is specified in a one-byte length-field at the beginning of each record. The record-length byte indicates the entire length of the record, **including** the length-byte. This can be any value from Ø to 255. A value of 1 can be used to mean an empty record (e.g., a blank line in an ASCII text file).



A length-byte value of 2 indicates that the record contains 1 byte of data:



A length-byte value of 16 indicates that the record contains 15 bytes of data:



RECORD NUMBERS

Records are numbered from Ø (beginning of file) to 16,777,214.

A disk file can also contain up to 16,777,216 bytes of storage, however, your storage medium may not be capable of storing this much information.

To determine the number of records a file will hold, use the following formula:

16777216 / logical record length = number of records

For example:

16777216 / 38 = 441,5Ø5 records

Example

If a three megabyte file (possible with hard disk) is opened with a record length of 3 bytes, it would hold approximately 1,ØØØ,ØØØ records.

RECORD PROCESSING CAPABILITIES

TRSDOS-16 allows both direct and sequential file access.

Direct access -- sometimes called "random access", allows you to process records in any sequence you specify.

Sequential access allows you to process records in sequence: Record N, N+1, N+2,.... With sequential access, you do not specify a record number; instead, TRSDOS-16 accesses the record following the last record processed, starting with record Ø.

FLR files may be opened as either direct access or sequential access.

VLR files can only be opened as sequential access. You cannot position to a specific record in the file, since the varying record lengths make it impossible to calculate the position of the VLRs.

The direct access SVC's are DIRRD (Direct-Read) and DIRWR (Direct-Write). Direct access SVC's always access the record you specify.

The sequential access SVC's are READNX (Read-Next) and WRITNX (Write-Next). Sequential access SVC's always access the record following the last record processed. (When the file is first opened, sequential processing starts with record Ø, however DIRWR and DIRRD can be used to position to the EOF, end of file.)

EXAMPLES WITH FIXED LENGTH FILES

Assume you have a Fixed Length Record file currently open. Here are some typical sequences you can accomplish via the file processing routines.

1. Read and/or write records in the file -- in any order
This is done using DIRRD and DIRWR SVC's. You could read record 5, write at end of file, read record 3, write record 3, etc.

2. Sequential Read (or Write) beginning anywhere in the file.
First you would do a direct-read to the record where you want to start reading or writing. After that, you would do sequential reads or writes until done (READNX and WRITNX).

3. Sequential Write starting at end of file.
First do a direct-write to the end of the file. Then do sequential writes until done.

4. Determine the number of records in a file.
First do a direct-read to end of file, then use the LOCATE routine to get the current record number, which now equals number of records +1.

EXAMPLES WITH VARIABLE LENGTH RECORDS

Here are examples of ways to read or write VLR files:

1. Start reading or writing sequentially at first record
Open the file and start reading or writing sequentially until done.

2. Sequential Write starting at end of file
First do a direct-write to the end of the file. Then do sequential writes until done.

Note: Whenever you write to a VLR, the end of the file is **automatically** reset to the last record you write.

Also, you cannot update a VLR file directly. You must read in the file, update it and output the updated information to a new VLR file.

SUPERVISOR CALLS

Supervisor Calls (SVCs) are operating system routines available to any user program. These routines alter certain system functions and conditions, provide file access, and perform I/O to the keyboard, video display, and printer.

The available TRSDOS-16 SVCs are:

KEYBOARD SVCs

KBCHAR
KBINIT
KBLINE
VIDKEY

VIDEO SVCs

CURSOR
VDCHAR
VDINIT
VDLINE

PRINTER SVCs

PRCHAR
PRCTRL
PRINIT
PRLINE

SYSTEM CONTROL SVCs

CLRXIT HLDKEY
DATE JP2DOS
DOSCMD SETBRK
ERRMSG SETTRP
ERROR VERSION

MISCELLANEOUS SVCs

DEBUG MOUNT
DISMOUNT MOVBUF
EXECUTE RESET

FILE ACCESS SVCs

CLOSE LOCATE
CLOSEF OPEN
DIRRD OPENDO
DIRRW READNX
DUMP RENAME
KILL UNLOCK
LOAD WRITNX

COMMUNICATIONS SVCs

ACTL BCTL
ARCV BRCV
ATX BTX
RS232C

Each SVC has a function code which you use to call it. These codes range from Ø to 512.

SVC BLOCK

To use an SVC, you must assign it an area in memory called a BLOCK. You use this BLOCK to pass parameters to and from the SVC.

The SVC block is a maximum of 16 words long. Each word consists of two bytes.

These words are used to store parameters which you want to pass to or from the SVC. They are addressed by their byte-offset number, an even number ranging from Ø to 32. Some SVC's will use only one byte of a word. In this case the byte will be located in the low order 8-bits (or byte-offset + 1). The high order 8-bits must contain zeroes.

Certain parameters, such as memory addresses, will use two consecutive words of storage (called a long word).

You can store an SVC BLOCK anywhere in memory; however, there must be a minimum of 32 bytes available after the starting address of the block, although the block may occupy less than last 32 bytes.

Figure 5 is a sample SVC BLOCK. (DIRWR SVC -- write a record out to disk.)

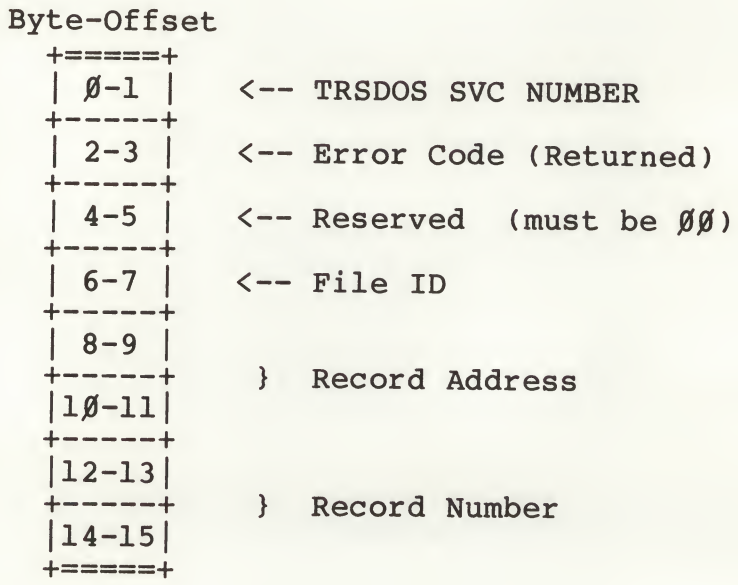


Figure 5. Sample SVC BLOCK

In this illustration, each box represents a word. The numbers within the boxes are the byte-offset numbers for addressing the SVC BLOCK.

The first number (the even number) is used to reference the offset. For example, byte-offset Ø-1 contains the SVC number, but is addressed as offset Ø.

Byte-offsets 16 through 31 are not used for this SVC, and need not be present.

The first three words of every SVC contain the same information:

Ø-1 TRSDOS Function Code Number (for example, 35)
2-3 On Error, returns the Error Code
4-5 Reserved -- Must contain zero

If an area within a block is marked RESERVED, you must set it to zero or a parameter error will occur.

CALLING PROCEDURE

To call a TRSDOS-16 SVC:

1. Load the address of the desired SVC BLOCK into register AØ.

2. Execute a BRK #Ø instruction.

3. Upon return from an SVC, you must specifically TEST for for an error condition if you want to see if one exists.

During the execution of an SVC, the SVC processor does not alter any registers, nor does it alter any status bits upon return.

Figure 6 is part of a sample program for the DIRWR SVC. We'll look at it closely to see one of the ways to load a SVC BLOCK for execution.

This sample assumes that the File Identification Number was previously stored in register A1, Record Address in register A2, and Record Number in register A3.

```
            .
            .
WRITE       LDA         .AØ,SVC BLOCK              * line 1
            MOVW        @AØ,#DIRWR SVC NUMBER      * line 2
            STW         A1,6@AØ                    * line 3
            STL         A2,8@AØ                    * line 4
            STL         A3,12@AØ                   * line 5
            BRK         #Ø                         * line 6
            TESTW       2@AØ                       * line 7
            BNE         ERROR                      * line 8
            .
            .
            .
SVC BLOCK
     RDATAB             32,Ø                       * line 9
DIRWR SVC NUMBER
            EQUW        44                         * line 1Ø
```

Figure 6. Sample Program

In this program, line 1 loads the address of the SVC BLOCK into register AØ.

Line 2 moves the function code number to byte-offset Ø and line 3 stores the file identification number in byte-offset 6.

Lines 4 and 5 store the Record Address and the Record Number in byte-offsets 8 and 12, respectively. Each is a long word.

Line 6 executes the SVC.

Line 7 tests for an error returned at offset 2.

If there is an error, the return is non-zero and line 8 either branches to an ERROR handling subroutine elsewhere in the program or executes the next line of the program.

Line 9 defines the SVC BLOCK to 32 bytes of zeroes.

And line 1Ø equates the SVC NUMBER to 44.

PROGRAMMING WITH USER INTERRUPTS

TRSDOS-16 allows user-programmed interrupts as described under SETBRK and SETTRP. When the interrupt is received (that is, when the <BREAK> key is pressed or a system trap is taken), control transfers to your interrupt handling routine.

When SETBRK terminates, or when an intercepted interrupt occurs, the following takes place:

1. The current User PC and status registers are pushed on the USER stack.

2. A branch is taken to the user-specified address.

Before doing any processing, you should save any registers you plan to alter. When processing is complete, execute an RTR instruction (return with restore) to return to the interrupted program.

A fatal error occurs if TRSDOS-16 cannot push three words onto the user stack, i.e., if the stack pointer contains a H'4 or less.

NOTE: Interrupt handlers entered this way are also subject to interrupts.

CONVERSION OF RELOCATABLE TO REAL ADDRESSES

If a byte-offset of an SVC 'points' to a buffer in user memory, the following restrictions apply:

All buffers are checked against their maximum possible sizes. It is not permissible to have a 256-byte buffer beginning 25 bytes from the end of memory. The check is made in this way:

1. DATE SVC buffer requires 26 bytes
2. ERRMSG SVC buffer requires 8Ø bytes
3. OPEN SVC parameter list requires 5 bytes

4. Any call passing a filespec (KILL, RENAME, etc.) requires a 34-byte buffer
5. Any Disk I/O call passing a buffer address is checked for a 256-byte buffer
6. Any miscellaneous I/O calls (PRLINE, VDLINE, etc.) passing a fixed length buffer is checked for 256 bytes.

ACTL **Function Code 1ØØ**
Control Channel A

Performs control functions on serial channel A and then returns its status.

TRSDOS-16 sets up A/BRCV, A/BTX and A/BCTL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

Entry Conditions

Byte-offset	Ø-1	1ØØ
	6-7	option switch

Exit Conditions

Byte-offset	2-3	error code
	8-9	communications status code

Valid option switch settings are:

option switch	Meaning
Ø	Get status of serial channel into the Status Value
1	Get received buffer count into the Status Value
2	Turn on Request to Send (RTS) signal
3	Turn off RTS
4	Start transmission of a BREAK sequence
5	Stop transmission of a BREAK sequence
6	Clears receive buffer
7	Reset SIO Error condition

The 8-bit grouping of flags returned in the communications status code are:

Bit	Meaning
Ø	Clear to Send not present
1	Unused
2	Transmitter busy
3	Modem data carrier not present
4	Parity error in byte now being received
5	Data overflow due to a byte now being received
6	Framing error in a byte now being received
7	Break Sequence now being received

Example

```
            .
            .
ACTL        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#ACTL SVC NUMBER
            MOVW        6@AØ,#OPTION SWITCH
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
ACTL SVC NUMBER
            EQUW        1ØØ
OPTION SWITCH
            EQUW        Ø
```

ARCV **Function Code 96**
Channel A Receive

Returns a single character from serial channel A.

TRSDOS-16 sets up A/BRCV, A/BTX and A/BCTL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

INPUT BUFFER

Each channel (A and B) has its own internal 16-character receive buffer to reduce overruns when receiving data at high speeds. This buffer is a First-In, First-Out buffer (FIFO) and is established when the channel is initialized.

When a character is received by the ARCV and BRCV SVC's, it is stored in this buffer along with its status when it was received.

Each time a new character is received into the buffer the character at the top of the buffer (oldest character) is pushed into the character returned byte-offset and its status (stored in the buffer when it was received) is pushed into the communications status code byte-offset.

An overrun occurs only when the 17th character is received into the already-full buffer. The 16th character is replaced with the 17th. When the character that caused the overrun is retrieved into the character returned byte-offset, an overrun will be indicated in communication status code.

If there are no characters "waiting" in the buffer, the communications status code will reflect the current status of the serial interface.

Entry Conditions

Byte-offset	Ø-1	96

Exit Conditions

Byte-offset		
	2-3	error code
	6-7	character returned (if any)
	8-9	communications status code
	1Ø-11	character received status
		Ø = character received
		non-zero = character not received

The 8-bit grouping of flags returned in the communications status code are:

Bit	Meaning
Ø	Clear to Send not present
1	Unused
2	Transmitter busy
3	Modem data carrier not present
4	Parity error in byte now being received
5	Data overflow due to a byte now being received
6	Framing error in a byte now being received
7	Break Sequence now being received

Example

```
            .
            .
ARCV        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#ARCV SVC NUMBER
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
ARCV SVC NUMBER
            EQUW        96
```

ATX **Function Code 97**
Channel A Transmit

Outputs a single character to serial channel A.

TRSDOS-16 sets up A/BRCV, A/BTX and A/BCTL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

Data bytes will be transmitted even if no carrier is present. You must check the communication status code and character transmitted status for error conditions.

Entry Conditions

Byte-offset	Ø-1	97
	6-7	character to be sent

Exit Conditions

Byte-offset	2-3	error code
	8-9	communications status code
	1Ø-11	character transmitted status Ø = character transmitted non-zero = character not transmitted

The 8-bit grouping of flags returned in the communications status code are:

Bit	Meaning
Ø	Clear to Send not present
1	Unused
2	Transmitter busy
3	Modem data carrier not present
4	Parity error in byte now being received *
5	Data overflow due to a byte now being received *
6	Framing error in a byte now being received *
7	Break Sequence now being received *

* Bits 4 - 7 are only used when the character was not sent

Example

```
            .
            .
ATX         LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#ATX SVC NUMBER
            MOVW        6@AØ,#CHAR TO SEND
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
ATX SVC NUMBER
            EQUW        97
CHAR TO SEND
            TEXT        'G
```

BCTL **Function Code 1Ø1**
Control Channel B

Performs control functions on serial channel B and then returns its status.

Entry Conditions

Byte-offset	Ø-1	1Ø1
	6-7	option switch

Exit Conditions

Byte-offset	2-3	error code
	8-9	comunication status code

Valid option switch settings are:

Option Switch	Meaning
Ø	Get status of serial channel into the Status Value
1	Get received buffer count into the Status Value
2	Turn on Request to Send (RTS) signal
3	Turn off RTS
4	Start transmission of a BREAK sequence
5	Stop transmission of a BREAK sequence
6	Clears receive buffer
7	Reset SIO Error condition

The 8-bit grouping of flags returned in the communications status code are:

Bit	Meaning when set
Ø	Clear to Send not present
1	Unused
2	Transmitter busy
3	Modem data carrier not present

Bit	Meaning when set
4	Parity error in byte now being received
5	Data overflow due to a byte now being received
6	Framing error in a byte now being received
7	Break Sequence now being received

Example

```
            .
            .
BCTL        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#BCTL SVC NUMBER
            MOVW        6@AØ,#OPTION SWITCH
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
BCTL SVC NUMBER
            EQUW        1Ø1
OPTION SWITCH
            EQUW        Ø
```

BRCV **Function Code 98**
Channel B Receive

Returns a single character from serial channel B.

TRSDOS-16 sets up A/BRCV, A/BTX and A/BCTL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

INPUT BUFFER

Each channel (A and B) has its own internal 16-character receive buffer to reduce overruns when receiving data at high speeds. This buffer is a First-In, First-Out buffer (FIFO) and is established when the channel is initialized.

When a character is received by the ARCV and BRCV SVC's, it is stored in this buffer along with its status when it was received.

Each time a new character is received into the buffer the character at the top of the buffer (oldest character) is pushed into the character returned byte-offset and its status (stored in the buffer when it was received) is pushed into the communications status code byte-offset.

An overrun occurs only when the 17th character is received into the already-full buffer. The 16th character is replaced with the 17th. When the character that caused the overrun is retrieved into the character returned byte-offset, an overrun will be indicated in communication status code.

If there are no characters "waiting" in the buffer, the communications status code will reflect the current status of the serial interface.

Entry Conditions

Byte-offset	Ø-1	98

Exit Conditions

Byte-offset		
	2-3	error code
	6-7	character returned (if any)
	8-9	communications status code
	1Ø-11	character received status Ø = character received non-zero = character not received

The 8-bit grouping of flags returned in the communication status code are:

Bit	Meaning
Ø	Clear to Send not present
1	Unused
2	Transmitter busy
3	Modem data carrier not present
4	Parity error in byte now being received
5	Data overflow due to a byte now being received
6	Framing error in a byte now being received
7	Break Sequence now being received

Example

```
            .
            .
BRCV        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#BRCV SVC NUMBER
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
BRCV SVC NUMBER
            EQUW        98
```

BTX **Function Code 99**
Channel B Transmit

Outputs a single character to serial channel B.

TRSDOS-16 sets up A/BRCV, A/BTX and A/BCTL when you initialize channel A/B with RS232C. If you call any of these routines while the channel is not initialized (active), you will get an error return code of 1 (no function code exists).

Data bytes will be transmitted even if no carrier is present. You must check the communication status code and character transmitted status for error conditions.

Entry Conditions

Byte-offset	Ø-1	99
	6-7	character to be sent

Exit Conditions

Byte-offset	2-3	error code
	8-9	communications status code
	1Ø-11	character transmitted status Ø = character transmitted non-zero = character not transmitted

The 8-bit grouping of flags returned in the communications status code are:

Bit	Meaning
Ø	Clear to Send not present
1	Unused
2	Transmitter busy
3	Modem data carrier not present
4	Parity error in byte now being received *
5	Data overflow due to a byte now being received *
6	Framing error in a byte now being received *
7	Break Sequence now being received *

* Bits 4 - 7 are only used when the character was not sent

Example

```
            .
            .
BTX         LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#BTX SVC NUMBER
            MOVW        6@AØ,#CHAR TO SEND
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
BTX SVC NUMBER
            EQUW        99
CHAR TO SEND
            TEXT        'G
```

CLOSE **Function Code 42**
Close Disk Files

Terminates output to a specified file. This SVC first writes all unsaved data to disk, then updates the directory.

Entry Conditions

Byte-offset	Ø-1	42
	6-7	file identification number

Exit Conditions

Byte-offset	2-3	error code

The file identification number is a unique number assigned to each file when it is opened.

Example

In this example the file identification number was previously stored in register A1.

```
            .
            .
CLOSE       LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#CLOSE SVC NUMBER
            STW         .A1,6@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
     RDATAB             32,Ø
CLOSE SVC NUMBER
     EQUW               42
```

CLOSEF **Function Code 133**
Close Files

Closes all open files, except for a currently executing DO-file, SPOOL file, or a file opened by OPENDO.

Entry Conditions

Byte-offset	Ø-1	133

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
CLOSEF      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#CLOSEF SVC NUMBER
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
             .
             .
SVC BLOCK
            RDATAB      32,Ø
CLOSEF SVC NUMBER
            EQUW        265
```

CLRXIT **Function Code 257**
Clear User Memory and Jump to TRSDOS-16

Clears (writes binary zeroes) to user memory, then gives control to TRSDOS-16 Ready. If used with a DO-File, control proceeds to the next command in the DO-File.

Entry Conditions

Byte-offset	Ø-1	257

Exit Conditions

Byte-offset	2-3	error code

Example

```
          .
          .
CLRXIT    LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#CLRXIT SVC NUMBER
          BRK       #Ø
          .
          .
SVC BLOCK
     RDATAB         32,Ø
CLRXIT SVC NUMBER
     EQUW           257
```

CURSOR **Function Code 1Ø**
Position Cursor

Positions the cursor to specified screen coordinates. This routine treats ROW and COLUMN as Modulo 24 and Modulo 8Ø, respectively.

Entry Conditions

Byte-offset	Ø-1	1Ø
	6-7	row to position cursor
	8-9	column to position cursor

Exit Conditions

Byte-offset	2-3	error code

Example

```
          .
          .
CURSOR    LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#CURSOR SVC NUMBER
          MOVW      6@AØ,#ROW POSITION
          MOVW      8@AØ,#COLUMN POSITION
          BRK       #Ø
          TESTW     2@AØ
          BNE       ERROR
          .
          .
SVC BLOCK
          RDATAB    32,Ø
CURSOR SVC NUMBER
          EQUW      1Ø
ROW POSITION
          EQUW      12
COLUMN POSITION
          EQUW      4Ø
```

DATE **Function Code 45**
Return Date String

Returns the time and date as a 26-byte ASCII string with eight fields.

CONTENTS OF TIME/DATE STRING (SAMPLE)

```
|S A T |A P R| 2 8 |1 9 7 9|1 1 8 |1 3 . 2 Ø . 4 2| 4| 5 |
```

```
   |      |     |       |      |           |        |   |
NAME OF  MON. DAY OF  YEAR  DAY OF      TIME     MON.  DAY
 DAY           MON.        YEAR                  #   OF WEEK
```

For example:

SATAPR28197911813.2Ø.42Ø45

represents the data "Saturday, April 28, 1979, the 118th day of the year, 13:2Ø:42 hours, the fourth month of the year, the fifth day of the week".

Monday is considered day Ø. The date calculations are based on the Julian Calendar.

Entry Conditions

Byte-offset	Ø-1	45
	6-9	address of 26-byte buffer

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
DATE        LDA       .AØ,SVC BLOCK
            MOVW      @AØ,#DATE SVC NUMBER
            LDA       .A1,DATE BUFFER
            STL       .A1,6@AØ
            BRK       #Ø
            TESTW     2@AØ
```

```
            BNE         ERROR
            .
            .
            .
SVC BLOCK
     RDATAB             32,Ø
DATE SVC NUMBER
     EQUW               45
DATE BUFFER
     RDATAB             26,Ø
```

DEBUG **Function Code 259**
Load the Debugger

Turns the debugger on and off. This call is valid only if DEBUG was configured. See Appendix B of this section for more information on the Configuration Command File.

Entry Conditions

Byte-offset	Ø-1	259
	6-7	function code

Exit Conditions

Byte-offset	2-3	error code

Valid function codes are:

FUNCTION CODE	FUNCTION
Ø	Turns DEBUG off
1	Turns DEBUG on
2	Enters DEBUG

If you use function code 2, the Debugger will load with the PC register set to the current instruction address plus two.

Example

```
          .
          .
DEBUG     LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#DEBUG SVC NUMBER
          MOVW      6@AØ,#DEBUG ON
          BRK       #Ø
          TESTW     2@AØ
          BNE       ERROR
          .
          .
```

```
SVC BLOCK
          RDATAB     32,Ø
DEBUG SVC NUMBER
          EQUW       259
DEBUG ON
          EQUW       1
```

DIRRD **Function Code 35**
Direct Read

Reads a specified record of an FLR (fixed-length record) file.

If you have a VLR (variable-length record) file, you can use DIRRD to read only the first record (Ø) or the end of file (-1).

Entry Conditions

Byte-offset	Ø-1	35
	6-7	file identification number
	8-11	record buffer address
	12-15	record number Ø = position to beginning of file H'FFFFFFFF = position to end of file
	16	record lock flag

Exit Conditions

Byte-offset	2-3	error code

The file identification number is a unique number assigned to each file when it is opened.

The record buffer address is a 32 bit address that points to the beginning of a 256-byte Record Buffer; that is where the record will be placed after the disk read.

The record number is a 32 bit number specifying the desired record number.

If the record lock flag is non-zero, the specified record remains locked until the user program performs an UNLOCK on the record.

If the record lock flag is zero, the record is not locked.

Example

Before executing this program, store the File Identification Number in register A1 and the Record Number in register A3.

```
            .
            .
READ        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#DIRRD SVC NUMBER
            STW         .A1,6@AØ
            LDA         .A2,RECORD BUFFER
            STL         .A2,8@AØ
            STL         .A3,12@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
            .
SVC BLOCK
            RDATAB      32,Ø
DIRRD SVC NUMBER
            EQUW        35
RECORD BUFFER
            RDATAB      256,Ø
```

DIRWR **Functon Code 44**
Direct Write

Writes a specified record in an FLR (fixed length record) file.

With a VLR (variable length record) files, you can use DIRWR to write only the first record (Ø) or the end of the file (-1).

Entry Conditions

Byte-offset	Ø-1	44
	6-7	file identification number
	8-11	record buffer address
	12-15	record number
		Ø = position to beginning of file
		H'FFFFFFFF = position to end of file

Exit Conditions

Byte-offset	2-3	error code

The file identification number is a unique number assigned to each file when it is opened.

The record buffer address is a 32-bit address that points to the beginning of a 256-byte Record Buffer; that is where the record will be placed after the disk read.

The record number is a 32-bit number specifying the desired record number.

Example

Before executing this program, store the Identification Number in register A1, Record Address in register A2, and Record Number in register A3.

```
            .
            .
WRITE       LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#DIRWR SVC NUMBER
```

```
            STW         .A1,6@AØ
            LDA         .A2,RECORD BUFFER
            STL         .A2,8@AØ
            STL         .A3,12@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
            .
SVC BLOCK
     RDATAB             32,Ø
DIRWR SVC NUMBER
            EQUW        44
RECORD BUFFER
            RDATAB      256,Ø
```

DISMOUNT **Function Code 139**
Dismount Disk

Logically disconnects a MOUNTed disk device. This command is valid for drives Ø-3 and 5-7 for hard disk users and drives 1-3 for floppy disk users. The system disk may not be DISMOUNTed.

A DISMOUNT/MOUNT sequence is required for each diskette swap. (see Chapter 1)

You should make sure all files are closed before executing a DISMOUNT/MOUNT sequence

Entry Conditions

Byte-offset	Ø-1	139
	6-9	device name

Exit Conditions

Byte-offset	2-3	error code

Valid device names are:

FDØØ = floppy drive Ø
FDØ1 = floppy drive 1
FDØ2 = floppy drive 2
FDØ3 = floppy drive 3
HDØØ = hard drive 4
HDØ1 = hard drive 5
HDØ2 = hard drive 6
HDØ3 = hard drive 7

Example

```
          .
          .
DISMNT    LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#DISMNT SVC NUMBER
          LDA.      .A1,DEVICE NAME
          MOVL      6@AØ,@A1
          BRK       #Ø
          TESTW     2@AØ
          BNE       ERROR
          .
          .
```

```
SVC BLOCK
          RDATAB     32,Ø
DISMNT SVC NUMBER
          EQUW       139
DEVICE NAME
          TEXT       'FDØ1'
```

DOSCMD **Function Code 27Ø**
Execute TRSDOS-16 Command

Passes a command string to the TRSDOS-16 Ready mode for execution. After execution is complete, control returns to TRSDOS-16 Ready.

This command alters the buffer used by the MOVBUF SVC.

When chaining programs, do not use DOSCMD to execute the programs. Use EXECUT instead.

Entry Conditions

Byte-offset	Ø-1	27Ø
	6-7	length of string
	8-11	address of string

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
DOSCMD      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#DOSCMD SVC NUMBER
            MOVW        6@AØ,#STRING LENGTH
            LDA         .A1,STRING
            STL         .A1,8@AØ
            BRK         #Ø
             .
             .
SVC BLOCK
            RDATAB      32,Ø
DOSCMD SVC NUMBER
            EQUW        27Ø
STRING LENGTH
            EQUW        5
STRING
            TEXT        'DIR 3'
```

DUMP **Function Code 13Ø**
Dump Memory to Disk

Writes a 68ØØØ format program from memory to diskette. The Dump replaces any existing file with the same name.

Entry Conditions

Byte-offset	Ø-1	13Ø
	6-9	filespec address
	1Ø-13	start dump address
	14-17	end dump address
	18-21	relocation address
	22-25	transfer Address

Exit Conditions

Byte-offset	2-3	error code

filespec address specifies the memory location of a standard TRSDOS-16 file specification. The filename is an ASCII string terminated by a carriage return.

start dump address and end dump address are the beginning and ending locations in memory where the program resides.

relocation address is the starting location where you want the file to load. If you don't want to relocate the file, set the relocation address at Ø.

transfer address is the program entry point (the address of the first instruction to execute). If you specify a relocation address, this address offsets the transfer address.

Note: all addresses must be an even value.

Example

```
          .
          .
DUMP      LDA       .AØ,SVC BLOCK
          MOVW      @AØ,DUMP SVC NUMBER
```

```
            LDA         .A1,FILE NAME
            STL         .A1,6@AØ
            MOVL        1Ø@AØ,#START AND TRANS ADDRESS
            MOVL        14@AØ,#END DUMP ADDRESS
            MOVL        18@AØ,#RELOCATION ADDRESS
            MOVL        22@AØ,#START AND TRANS ADDRESS
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
              .
              .
SVC BLOCK
            RDATAB      32,Ø
DUMP SVC NUMBER
            EQUW        13Ø
FILE NAME
            TEXT        'TEST/SRC'
            DATAB       H'ØD
START AND TRANS ADDRESS
            EQUL        H'6ØØØØ
END DUMP ADDRESS
            EQUL        H'6ØØØØ
RELOCATION ADDRESS
            EQUL        Ø
```

ERRMSG **Function Code 52**
Error Message

In response to a requested error number, this routine returns an 8Ø-byte descriptive error message to a specified buffer area.

Entry Conditions

Byte-offset	Ø-1	52
	6-7	error number
	8-11	message buffer address

Exit Conditions

Byte-offset	2-3	error code

The error number range is Ø to 255, inclusive.

Example

```
            .
            .
ERRMSG      LDW         .A1,2@AØ
            LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#ERRMSG SVC NUMBER
            STW         .A1,6@AØ
            LDA         .A1,MESSAGE BUFFER
            STL         .A1,8@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         JP2DOS
            CALL        VIDEO PRINT LINE ROUTINE
            .
            .
SVC BLOCK
            RDATAB      32,Ø
ERRMSG SVC NUMBER
            EQUW        52
MESSAGE BUFFER
            RDATAB      8Ø,Ø
```

ERROR **Function Code 39**

Display Error Number

Displays the message ERROR followed by the specified error code at the current cursor position.

Entry Conditions

Byte-offset	Ø-1	39
	6-7	error number

Exit Conditions

Byte-offset	2-3	error code

The error number is a Ø - 255 code of the message you want displayed.

Example

```
            .
            .
ERROR       LDW         .A1,2@AØ
            LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#ERROR SVC NUMBER
            STW         .A1,6@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         JP2DOS
             .
             .
SVC BLOCK
            RDATAB      32,Ø
ERROR SVC NUMBER
            EQUW        39
```

EXECUT **Function Code 263**
Execute Program

Begins program execution in user memory.

Entry Conditions

Byte-offset	Ø-1	263
	6-7	ØØ (Reserved)
	8-11	filespec address

Exit Conditions

Byte-offset	2-3	error code

The filespec address is the 32-bit address of a valid TRSDOS-16 filespec terminated by a carriage return.

Example

```
            .
            .
EXECUTE     LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#EXECUTE SVC NUMBER
            MOVW        6@AØ,#ØØ
            LDA         .A1,FILENAME
            STL         .A1,8@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         JP2DOS
            .
            .
SVC BLOCK
            RDATAB      32,Ø
EXECUTE SVC NUMBER
            EQUW        263
FILENAME
            TEXT        'TEST/CMD'
            DATAB       H'ØD
```

HLDKEY **Function Code 29**
Enable Hold Key

Suspends and restarts terminal output whenever you press the <HOLD> key.

This SVC must first be enabled. Then you must periodically call HLDKEY to check if the <HOLD> key has been depressed. This places the program in control of the pause checking.

Note that execution of certain DOSCMDs might alter the HLDKEY status. This will happen if the TRSDOS-16 command issues HLDKEY requests.

Entry Conditions

Byte-offset	Ø-1	29
	6-7	function code

Exit Conditions

Byte-offset	2-3	error code

The HLDKEY function codes are:

Code	Meaning
Ø	Turn off HOLD processor. Pressing <HOLD> generates H'ØØ'.
1	Turn on HOLD processor. <HOLD> key doesn't generate keyboard data, it is intercepted by TRSDOS-16.
2	Check for <HOLD> key. If pressed, wait until pressed again.

Example

```
          .
          .
HLDKEY    LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#HLDKEY SVC NUMBER
          MOVW      6@AØ,#HOLD KEY FUNCTION CODE
```

```
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR

* CHECK TO SEE IF HOLD KEY PRESSED
            LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#HLDKEY SVC NUMBER
            MOVW        6@AØ,#CHECK IF HOLD KEY WAS PRESSED
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
              .
              .
SVC BLOCK
            RDATAB      32,Ø
HDLKEY SVC NUMBER
            EQUW        29
HOLD KEY FUNCTION CODE
            EQUW        1
CHECK IF HOLD KEY PRESSED
            EQUW        2
```

JP2DOS **Function Code 264**
Jump to TRSDOS-16

After closing all open files and performing system housekeeping, returns control to TRSDOS-16 Ready.

If a program under the control of a DO-file executes this routine, control proceeds to the next command in the DO-file.

Entry Conditions

Byte-offset	Ø-1	264
	6-7	ØØ (Reserved)

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
JP2DOS      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#JP2DOS SVC NUMBER
            BRK         #Ø
            .
            .
SVC BLOCK
            RDATAB      32,Ø
JP2DOS SVC NUMBER
            EQUW        264
```

KBCHAR **Function Code 4**
Keyboard Character

Checks the keyboard for a new character entry.

If characters are present in the key-ahead buffer, the first character in the buffer will be returned in the returned character byte-offset.

The <BREAK> key is masked from the user -- it will never be returned, since it is intercepted by the System. If a SETBRK routine is enabled, control passes to the processing program (see SETBRK) whenever <BREAK> is pressed. Otherwise, control will pass to TRSDOS-16 Ready.

Entry Conditions

Byte-offset	Ø-1	4
	6-7	wait for character flag

Exit Conditions

Byte-offset	2-3	error code
	6-7	character present flag
	8-9	returned character

If you want the SVC to return only when a character is detected, set the wait for character flag to non-zero. A zero in this byte-offset causes the SVC to return, with or without a character, immediately after the keyboard buffer is checked.

A character present flag of non-zero means the routine has returned with a valid ASCII character in the returned character position.

If it returns without a character, (when the SVC is set not to wait for a character) the routine clears the character present flag, but doesn't change the contents of the returned character byte-offset.

Example

This program returns the character most recently pressed and then calls PRINT CHARACTER ROUTINE, which uses either the VDCHAR or PRCHAR routines to print it.

```
            .
            .
KBCHAR      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#KBCHAR SVC NUMBER
            MOVW        6@AØ,#WAIT FOR CHARACTER FLAG
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            CALL        PRINT CHARACTER ROUTINE
            .
            .
SVC BLOCK
            RDATAB      32,Ø
KBCHAR SVC NUMBER
            EQUW        4
WAIT FOR CHARACTER FLAG
            EQUW        1
```

KBINIT **Function Code 1**
Keyboard Initialize

Initializes the keyboard input driver. You might want to call this SVC before starting keyboard input to clear previous keystrokes and the key-ahead buffer. TRSDOS-16 does this automatically at start-up.

Entry Conditions

Byte-offset	Ø-1	1

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
KBINIT      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#KBINIT SVC NUMBER
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
KBINIT SVC NUMBER
            EQUW        1
```

KBLINE **Function Code 5**
Keyboard Line

Inputs a line from the keyboard to a buffer and echoes the line to the Display, starting at the current cursor position. As it receives and displays each character, the routine advances the cursor to the next position.

When you enter this routine, the input buffer contains a string of periods which it sends to the display. This string is for your convenience to indicate the length of the input field.

The keyboard line ends when you press <ENTER> or when the routine fills the input buffer. When you terminate line input, the routine always sends a carriage return and an erase-to-end-of-screen command to the Display. It stores a carriage return as a character input only if you actually press <ENTER>.

If you type a control code not listed below, the routine places it in the buffer and represents it on the display with the ± symbol.

KEY	HEX CODE	FUNCTION
BACKSPACE	Ø8	Backspaces the cursor and erases a character.
ENTER	ØD	Terminates line. Clears trailing periods on Display
CTRL W	17	Fills remainder of input buffer with blanks, blanks remainder of Display line
CTRL X	18	Fills remainder of input buffer with blanks, blanks to end of Display.
ESC	1B	Reinitializes input function by filling input buffer with periods and restoring cursor to original position.
<-	1C	Backspaces the cursor to allow editing of line. Does not erase characters.
->	1D	Advances the cursor to allow editing of line. Does not erase characters.

Entry Conditions

Byte-offset	Ø-1	5
	6-7	maximum number of characters to receive (Ø - 255)
	8-11	input buffer address

Exit Conditions

Byte-offset	2-3	error code
	12-13	actual number of characters input
	14-15	terminating character of input H'ØD if line was terminated with a carriage return, Ø if buffer was filled without a carriage return

Example

```
            .
            .
KBLINE      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#KBLINE SVC NUMBER
            MOVW        6@AØ,#MAXIMUM INPUT COUNT
            LDA         .A1,KEYBD BUFFER
            STL         .A1,8@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
KBLINE SVC NUMBER
            EQUW        5
MAXIMUM INPUT COUNT
            EQUW        8Ø
KEYBD BUFFER
            RDATAB      8Ø,Ø
```

KILL **Function Code 41**
Delete File from Directory

Deletes the specified file from the directory. A file must be closed for you to KILL it.

Entry Conditions

Byte-offset	Ø-1	41
	6-9	filespec address

Exit Conditions

Byte-offset	2-3	error code

The filespec address is the 32-bit address of a valid TRSDOS-16 filespec terminated by a carriage return.

Example

```
            .
            .
KILL        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#KILL SVC NUMBER
            LDA         .A1,FILE NAME
            STL         .A1,6@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
             .
             .
SVC BLOCK
            RDATAB      32,Ø
KILL SVC NUMBER
            EQUW        41
FILE NAME
            TEXT        'TEST/SRC'
            DATAB       H'ØD
```

LOAD **Function Code 131**

LOAD's a 68ØØØ program into user memory.

Entry Conditions

Byte-offset	Ø-1	131
	6-9	filespec address
	1Ø-13	low bound address (only if program is position independent, otherwise must be zero)
	14-17	high bound address (only if program is position independent, otherwise zero allows all of memory)

Exit Conditions

Byte-offset	2-3	error code
	1Ø-13	start load address
	14-17	address of last byte loaded
	18-21	transfer address

Supply a low bounds address and high bounds address only if your program does NOT have any absolute memory references.

filespec address is the 32-bit address of a valid TRSDOS-16 program filespec terminated by a carriage return.

low bounds address is the logical address you want the program loaded at. Specifying zero means the entire user memory is available. If this number is non-zero then the program MUST be position-independent.

high bounds address is the logical address of the last byte available for the program being loaded. Specifying zero allows entire user memory.

start load address is the location in memory of the first byte loaded.

address of last byte loaded is the highest address of the loaded program.

transfer address is the program entry point (address of first instruction to be executed). If the program is nonexecutable ("load-only"), the byte-offset contains a -1.

Note: all addresses must be an even value.

Example

```
            .
            .
LOAD        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#LOAD SVC NUMBER
            LDA         .Al,FILE NAME
            STL         .Al,6@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
LOAD SVC NUMBER
            EQUW        131
FILE NAME
            TEXT        'TEST/SRC'
            DATAB       H'ØD
```

LOCATE **Function Code 33**
Locate Record

Returns the number of the current record, i.e., the last record accessed. You can call this routine only with FLR (fixed-length Record) files.

Entry Conditions

Byte-offset	Ø-1	33
	6-7	file identification number

Exit Conditions

Byte-offset	2-3	error code
	8-11	record number

The file identification number is a unique number assigned to each file when it is opened.

The record number is a 32 bit number specifying the desired record number. The record number will be zero if the file was just opened.

Example

Prior to execution of this routine, store the file identification number in register A1.

```
            .
            .
LOCATE      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#LOCATE SVC NUMBER
            STW         .A1,6@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
LOCATE SVC NUMBER
            EQUW        33
```

MOUNT **Function Code 138**
Mount Device

Logically connects a device to the system. The system cannot access an unMOUNTed device. You must mount a diskette prior to any I/O attempts.

A DISMOUNT/MOUNT sequence is required for each floppy diskette swap.

You should make sure all files are closed before executing a DISMOUNT/MOUNT sequence.

Entry Conditions

Byte-offset	Ø-1	138
	6-9	device name

Exit Conditions

Byte-offset	2-3	error code

Valid device names are:

FDØØ = floppy drive Ø
FDØ1 = floppy drive 1
FDØ2 = floppy drive 2
FDØ3 = floppy drive 3
HDØØ = hard drive 4
HDØ1 = hard drive 5
HDØ2 = hard drive 6
HDØ3 = hard drive 7

Example

```
          .
          .
MOUNT     LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#MOUNT SVC NUMBER
          LDA       .A1,DEVICE NAME
          MOVL      6@AØ,@A1
          BRK       #Ø
          TESTW     2@AØ
          BNE       ERROR
          .
          .
```

```
SVC BLOCK
          RDATAB     32,Ø
MOUNT SVC NUMBER
          EQUW       138
DEVICE NAME
          TEXT       'FDØ1'
```

MOVBUF **Function Code 267**
Move Buffer

Either retrieves or stores an 8Ø-byte buffer outside the user's memory. This buffer serves two purposes:

1. To pass parameters when chaining two machine language programs.

   (When chaining programs, do not use DOSCMD to execute the programs. Use EXECUT instead.)

2. To retrieve a TRSDOS-16 command line. When TRSDOS-16 starts a user program, it stores the command line that invoked the job in this area.

   (When TRSDOS-16 stores a command line it terminates it with a carriage-return character. However, when MOVBUF retrieves a command line 8Ø characters long, it does not retrieve the carriage return, implied 81st character.)

There are two ways to alter this buffer:

1. A MOVBUF request from memory to buffer overwrites anything in the buffer. Note that 8Ø bytes (5Ø Hex bytes) are always moved.

2. A DOSCMD request does an implied move into the buffer before executing the request. Anything previously stored in the buffer is lost.

Entry Conditions

Byte-offset	Ø-1	267
	6-9	user buffer address
	1Ø-11	switch
		Ø = Retrieve
		non-zero = Store

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
MOVBUF      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#MOVBUF SVC NUMBER
            LDA         .Al,BUFFER
            STL         .Al,6@AØ
            MOVW        1Ø@AØ,#FETCH SWITCH
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
MOVBUF SVC NUMBER
            EQUW        267
BUFFER
            RDATAB      8Ø,Ø
FETCH SWITCH
            EQUW        Ø
```

OPEN **Function Code 4Ø**
Open File

Creates new files and opens existing files.

Once a given file has been opened, OPEN assigns a unique file identification number to the file. This number is used in all other file processing SVC's to designate that specific file.

Entry Conditions

Byte-offset	Ø-1	4Ø
	6-9	filespec address
	1Ø-13	parameter list address

Exit Conditions

Byte-offset	2-3	error code
	14-15	file identification number

The filespec address is the 32-bit address of a valid TRSDOS-16 filespec terminated by a carriage return.

The parameter list address is the address of a 5-bytes field (Ø-4) which contains:

Byte	Contents
ØØ	access code "R" Read access to the file "W" Read/Write (data files) "P" Read/Write access to Z8Ø program files
Ø1	record length in bytes
Ø2	file type "F" fixed "V" variable
Ø3	creation code (Ø-3 -- see table)
Ø4	user attribute byte

The creation code specifies the way TRSDOS-16 opens the file:

Code	Function
Ø	Open the file only if it already exists. Do not create a new file. The record length and end of file pointers are not reset. Exclusive access only.
1	Creates a new file. Does not open an existing file (returns an error if a file of the same name already exists on the specified drive). Resets the record length and end of file. Exclusive access only.
2	Open a new file. If a file of the same name already exists, overwrite it. Resets record length and end of file. Exclusive access only.
3	Opens an existing file for shared access. The file MUST already exist (same as mode Ø).

The user attribute byte allows you to give your own identification number to certain types of data files.

You can use Ø or any number from 32 - 255 for this value.

TRSDOS-16 will not examine this user attribute; it is solely for your convenience.

You can assign this user attribute only to files you open with a creation code of 1 or 2. Files opened with a creation code of Ø or 3 will retain the file's previously assigned user attribute. All files created with the CREATE command will have a user attribute value of zero.

The file identification number is a unique number assigned to each file when it is opened.

Example

```
            .
            .
OPEN        LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#OPEN SVC NUMBER
            LDA         .Al,FILE NAME
            STL         .Al,6@AØ
            LDA         .Al,PARAM LIST
            MOVW        @Al,#WRITE ACCESS
            MOVW        1@Al,#RECORD LENGTH
            MOVW        2@Al,#FIXED FILE
            MOVW        3@Al,#OPEN ONLY IF EXISTS
            MOVW        4@al,#USER ATTRIB
            STL         .Al,1Ø@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
OPEN SVC NUMBER
            EQUW        4Ø
FILE NAME
            TEXT        'TEST/SRC'
            DATAB       H'ØD
PARAM LIST
            RDATAB      5,Ø
WRITE ACCESS
            EQUB        'P
RECORD LENTGH
            EQUB        8Ø
FIXED FILE
            EQUB        'F
OPEN ONLY IF EXISTS
            EQUB        Ø
USER ATTRIB
            EQUB        66
```

OPENDO **Function Code 14Ø**
Open DO File

Creates a special file or opens an existing one.

OPENDO opens a special file that will not be closed by returning to TRSDOS-16 Ready or executing the CLOSEF SVC.

OPENDO is useful for chaining programs.

Once a given file has been opened, OPENDO assigns a unique file identification number to the file. This number is used in all other file processing SVC's to designate that specific file.

Only one file may be opened with OPENDO at a time. Therefore, this call cannot be made if a DO file is active.

Entry Conditions

Byte-offset		
	Ø-1	14Ø
	6-9	filespec address
	1Ø-13	parameter list address

Exit Conditions

Byte-offset		
	2-3	error code
	14-15	file identification number

The filespec address is the 32-bit address of a valid TRSDOS-16 filespec terminated by a carriage return.

The parameter list address is the address of a 5-bytes field (Ø-4) which contains:

Byte	Contents
ØØ	access code
	"R" Read access to the file
	"W" Read/Write (data files)
	"P" Read/Write access to Z8Ø program files

You may not execute the SPOOL command and a DO file at the same time. The first one executed will be given priority over the second.

Byte	Contents
Ø1	record length in bytes
Ø2	file type "F" fixed "V" variable
Ø3	creation code (Ø-3 -- see table)
Ø4	user attribute byte

The creation code specifies the way TRSDOS-16 opens the file:

Code	Function
Ø	Open the file only if it already exists. Do not create a new file. The record length and end of file pointers are not reset. Exclusive access only.
1	Creates a new file. Does not open an existing file (returns an error if a file of the same name already exists on the specified drive). Resets the record length and end of file. Exclusive access only.
2	Open a new file. If a file of the same name already exists, overwrite it. Resets record length and end of file. Exclusive access only.
3	Opens an existing file for shared access. The file MUST already exist (same as mode Ø).

The user attribute byte allows you to give your own identification number to certain types of data files.

You can use Ø or any number from 32 - 255 for this value.

TRSDOS-16 will not examine this user attribute; it is solely for your convenience.

You can assign this user attribute only to files you open with a creation code of 1 or 2. Files opened with a

creation code of Ø or 3 will retain the file's previously assigned user attribute. All files created with the CREATE command will have a user attribute value of zero.

The file identification number is a unique number assigned to each file when it is opened.

Example

```
            .
            .
OPENDO      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#OPENDO SVC NUMBER
            LDA         .Al,FILE NAME
            STL         .Al,6@AØ
            LDA         .Al,PARAM LIST
            MOVW        @Al,#WRITE ACCESS
            MOVW        1@Al,#RECORD LENGTH
            MOVW        2@Al,#FIXED FILE
            MOVW        3@Al,#OPEN ONLY IF EXISTS
            MOVW        4@al,#USER ATTRIB
            STL         .Al,1Ø@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
OPENDO SVC NUMBER
            EQUW        14Ø
FILE NAME
            TEXT        'TEST/SRC'
            DATAB       H'ØD
PARAM LIST
            RDATAB      5,Ø
WRITE ACCESS
            EQUB        'P
RECORD LENTGH
            EQUB        8Ø
FIXED FILE
            EQUB        'F
OPEN ONLY IF EXISTS
            EQUB        Ø
USER ATTRIB
            EQUB        66
```

PRCHAR **Function Code 18**
Print Character

Sends one character to the Printer's buffer.

Note: Most printers don't print until their buffer is filled or until they receive a carriage return. (See your printer's manual for information.)

Normally, TRSDOS-16 intercepts certain codes and does not send them directly to the printer. There are several ways to override some or all of these character translations. See PRINIT and PRCTRL SVCs.

While the serial printer option is selected, allowing printer output to Channel B, two INPUT characters are recognized from Channel B. These will affect all serial printer output operations:

ASCII Name	Hex Code	Result
DC3, "CTRL-S"	13	Pause Printing
DC1, "CTRL-Q"	11	Resume Printing

INTERCEPTED CODES

ASCII NAME	HEX CODE	RESULT TO PRINTER
Tab	Ø9	From one to eight spaces are sent to provide a tab function.
Vertical Tab	ØB	Same as form feed below.
Form Feed	ØC	TRSDOS-16 sends enough carriage returns or line feeds to the printer to advance the paper to the next top of form.

ASCII NAME	HEX CODE	RESULT TO PRINTER
Carriage Return	ØD	When the current line is empty, (no characters printed since the last carriage return or line feed), TRSDOS-16 translates this as a line feed to allow correct operation of Radio Shack printers. In the auto line feed mode, H'ØA' is sent after every H'ØD'.
Special	8D	TRSDOS-16 sends a carriage return to the printer. In the auto line feed mode, using this code allows you to send a carriage return without a line feed.

Entry Conditions

Byte-offset	Ø-1	18
	6-7	character to output

Exit Conditions

Byte-offset	2-3	error code

Example

Before executing this program, load register A1 with the character from byte-offset 8 of the KBCHAR SVC routine.

```
            .
            .
PRCHAR      LDW         .A1,8@AØ
            LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#PRCHAR SVC NUMBER
            STW         .A1,6@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
PRCHAR SVC NUMBER
            EQUW        18
```

PRCTRL **Function Code 95**
Control Printer Operation

Provides you with various printer options and lets you check the status of printer-related functions.

If you are using the spooler's capture function, the captured data is sent directly to the capture-file -- regardless of what control settings are selected. If you are using the spooler's background printing function, the printed data is interpreted according to the currently selected control settings. See SPOOL for details.

You can operate with two printers, by switching between the serial and parallel output modes. One printer can use the automatic control features of TRSDOS (auto form feed, etc.), while the other printer is controlled by the program. This is necessary since **only one set of control counters is maintained.**

Entry Conditions

Byte-offset	Ø-1	95
	6-7	option code (See Table 2)
	8-9	option value (applicable only with option codes 3 and 4)

Exit Conditions

Byte-offset	2-3	error code
	1Ø-11	physical page length
	12-13	logical page length
	14-15	logical line length
	16-17	number of characters printed on present line
	18-19	number of lines on present page

The option code is a 16-bit value used to select the print control option you wish.

The option value is used by option codes 3 and 4 which allow you to reset the current line count or the current character count. Put the new values in this byte-offset.

Available option codes are:

CODE	OPTION
Ø	Get printer status only (see Exit Conditions)
1	Select serial printer driver (you must initialize channel B first)
2	Select parallel printer driver
3	Reset current line count to the value contained in option value *
4	Reset current character count on current line to value contained in option value *
5	Begin transparent mode *
6	End transparent mode *
7	Begin dummy mode *
8	End dummy mode *
9	Begin auto line-feed after carriage return *
1Ø	End auto line-feed after carriage return *

* EXPLANATION OF OPTIONS

EXPLANATION OF OPTIONS

Serial/Parallel Printer Option

While the serial printer option is selected, allowing printer output to Channel B, two INPUT characters are recognized from Channel B. These will affect all serial printer output operations:

ASCII Name	Hex Code	Result
DC3, "CTRL-S"	13	Pause Printing
DC1, "CTRL-Q"	11	Resume Printing

Line count/Character count

TRSDOS-16 maintains a single line counter and a single character counter.

Transparent Mode

The transparent mode overrides all data translation. All data bytes go directly to the printer; TRSDOS-16 does not examine the contents or update the line and character counts.

Normally, TRSDOS-16 "intercepts" certain control characters and interprets them. In this way, TRSDOS-16 provides printer-related features which may not be available from the printer.

For example, tabs (H'Ø9') are intercepted so that TRSDOS-16 can send the appropriate number of spaces to provide the tab function. Form feeds (H'ØC' or H'ØB') are intercepted so that TRSDOS-16 may advance the paper to the top of the next form.

Special settings of the printer initialization values can override individual code translation. See SVC PRINIT for details.

Dummy Output Mode

The dummy mode "throws away" all printer output and returns with a "good" (Z flag set) return code. During dummy mode operation, the line count and character count remain unchanged.

Auto Line Feed

Normally, the TRSDOS-16 printer driver doesn't output line feeds after carriage returns because most Radio Shack printers do an automatic line-feed after every carriage return.

If your printer does not perform automatic line-feeds after carriage returns, you may enable the TRSDOS-16 auto line-feed function.

While the function is enabled, TRSDOS-16 outputs a line-feed after each carriage return. This is true in all modes, including the transparent mode.

Note: In the auto line-feed mode, you can send a carriage return with no line-feed by outputting the code H'8D'. If the printer can overprint, this code returns the carriage without advancing the paper.

PRECEDENCE OF OPTIONS

The priority of the available options are (in descending order):

Dummy Mode
Auto Line-Feed after Carriage Return
Transparent Mode
Normal Mode

Example

```
            .
            .
PRCTRL      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#PRCTRL SVC NUMBER
            MOVW        6@AØ,#GET STATUS OPTION
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
             .
             .
SVC BLOCK
            RDATAB      32,Ø
PRCTRL SVC NUMBER
            EQUW        95
GET STATUS OPTION
            EQUW        Ø
```

PRINIT **Function Code 17**
Printer Initialization

Initializes the printer driver.

It does **not** advance the printer paper and does **not** check printer status or availability. It will operate even if the printer is offline.

Entry Conditions

Byte-offset	Ø-1	17
	6-7	physical page length
	8-9	logical page length
	1Ø-11	logical line length

Exit Conditions

Byte-offset	2-3	error code

physical page length is the number of lines that can be printed on one page. It is normally 66.

logical page length is the number of lines you want printed on each page. It must be less than physical page length. After that number of lines has been printed, (logical page length < physical page length) TRSDOS-16 will send a top-of-page request to the printer (form feed).

logical line length is the maximum number of characters to be printed on each line. Once that number has been reached, the line will be printed and a new line is begun (wrap-around).

Notes:

1. If logical page length and physical page length are equal and non-zero, TRSDOS-16 will not do an end-of-page skip. It will translate form feed into the correct number of line feeds necessary to get to the top of the next page.

2. If either logical page length or physical page length are zero, the other **must** also be zero. When both are equal to zero, form feeds (H'ØC) and vertical tabs (H'ØB) are not translated into line feeds, but are sent directly to the printer.

3. If logical line length is zero, then tab characters (H'Ø9) will not be translated into spaces, but sent directly to the printer. TRSDOS-16 will still maintain an internal character count, with tabs counting as one character.

4. On Exit, current character count and line-in-page count are reset to zero.

5. TRSDOS-16 assumes the printer is at top-of-page when this SVC is executed.

6. DUMMY and TRANSPARENT modes are reset to NORMAL. AUTO-LINEFEED and DUAL options are unaffected by this SVC. (See PRCTRL)

7. When the system is powered-up or reset, the following defaults are assumed:

physical page length	=	66
logical page length	=	6Ø
logical line length	=	132

The other options selected during initialization are:

Auto-Line Feed	=	off
Dummy Mode	=	off
Transparent Mode	=	off

Example

```
PRINIT    LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#PRINIT SVC NUMBER
          MOVW      6@AØ,#PHY PAGE LENGTH
          MOVW      8@AØ,#LOG PAGE LENGTH
          MOVW      1Ø@AØ,#LOG LINE LENGTH
```

```
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR

              .
              .
SVC BLOCK
            RDATAB      32,Ø
PRINIT SVC NUMBER
            EQUW        17
PHY PAGE LENGTH
            EQUW        32
LOG PAGE LENGTH
            EQUW        3Ø
LOG LINE LENGTH
            EQUW        45
```

PRLINE **Function Code 19**
Print Line

Sends a line of characters to the printer's buffer. The line can include control characters as well as printable data.

See PRCHAR for a listing of intercepted codes.

Entry Conditions

Byte-offset	Ø-1	19
	6-7	length of line (Ø-255)
	8-9	terminator character to send after last character in buffer
	1Ø-13	address of buffer

Exit Conditions

Byte-offset	2-3	error code

Example

In this example, TRSDOS-16 retrieves the length of line, terminator character, and address of buffer from the KBLINE SVC routine.

```
            .
            .
            LDW       .A1,12@AØ  *Get length of line
            LDW       .A2,14@AØ  *Get terminator character
            LDA       .A3,KEYBD BUFFER   *Get buffer address
PRLINE      LDA       .AØ,SVC BLOCK
            MOVW      @AØ,#PRLINE SVC NUMBER
            STW       .A1,6@AØ
            STW       .A2,8@AØ
            STL       .A3,1Ø@AØ
            BRK       #Ø
            TESTW     2@AØ
            BNE       ERROR
            .
            .
```

```
SVC BLOCK
          RDATAB     32,Ø
PRLINE SVC NUMBER
          EQUW       19
```

READNX **Function Code 34**
Read Next Record

Reads the next record after the current record (the last record accessed). If you have just opened the file, READNX reads the first record.

Entry Conditions

Byte-offset	Ø-1	34
	6-7	file identification number
	8-11	record buffer address
	12-13	record lock flag

Exit Conditions

Byte-offset	2-3	error code

The file identification number is a unique number assigned to each file when it is opened.

The record buffer address is a 32-bit address that points to the beginning of a 256-byte Record Buffer; that is where the record will be placed after the disk read.

If the record lock flag is non-zero, the specified record remains locked until the user program performs an UNLOCK on the record.

If the record lock flag is zero, the record is not locked.

Example

Prior to this routine, store the file identification number in register A1.

```
            .
            .
READNX      LDA       .AØ,SVC BLOCK
            MOVW      @AØ,#READNX SVC NUMBER
            STW       .A1,6@AØ
            LDA       .A2,RECORD BUFFER AREA
            STL       .A2,8@AØ
```

```
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
              .
              .
SVC BLOCK
            RDATAB      32,Ø
READNX SVC NUMBER
            EQUW        34
RECORD BUFFER AREA
            RDATAB      256,Ø
```

RENAME **Function Code 47**
Rename File

Changes the name and/or extension of a file.

This SVC cannot change the password. If the old filespec is password protected, you must specify the password and the new filespec retains the same password.

The new filespec can't refer to an existing file.

Entry Conditions

Byte-offset	Ø-1	47
	6-9	old filespec address
	1Ø-13	new filespec address

Exit Conditions

Byte-offset	2-3	error code

The filespec address is the 32-bit address of a valid TRSDOS-16 filespec terminated by a carriage return.

Example

```
            .
            .
RENAME      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#RENAME SVC NUMBER
            LDA         .Al,OLD NAME
            STL         .Al,6@AØ
            LDA         .Al,NEW NAME
            STL         .Al,1Ø@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
RENAME SVC NUMBER
            EQUW        47
```

```
OLD NAME
          TEXT      'TEST/SRC'
          DATAB     H'ØD
NEW NAME
          TEXT      'PROGRAM/SRC'
          DATAB     H'ØD
```

RESET **Function Code 129**
Reset Memory

It is the same as pressing the RESET switch.

Reset will not close an OPENDO file.

Entry Conditions

Byte-offset	Ø-1	256
	6-7	ØØ (Reserved)

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
RESET       LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#RESET SVC NUMBER
            MOVW        6@AØ,#ØØ
            BRK         #Ø
            .
            .
SVC BLOCK
            RDATAB      32,Ø
RESET SVC NUMBER
            EQUW        129
```

RS232C **Function Code 55**
Initialize RS-232-C Channel

Sets up or disables either channel A or B.

This routine sets the standard RS-232C parameters, and defines a pair of supervisor calls for I/O to the specified channel. When you initialize Channel A, SVC's 96, 97, 1ØØ are defined; when you initialize Channel B, SVC's 98, 99, and 1Ø1 are defined. See ARCV, ATX, BRCV, BTX, ACTL and BCTL.

Entry Conditions

Byte-offset	Ø-1	55
	6-7	option value Ø = deactivate non-zero = activate
	8-11	parameter list address

Exit Conditions

Byte-offset	2-3	error code

Set the option value to Ø to deactivate the serial channel. Any non-zero value activates the channel. You must always deactivate the channel (turn it off) before you set the parameters.

The parameter list address is the address of a 6-byte field (Ø-5) which contains:

Byte	Contents
ØØ	Channel code 'A' or 'B' (ASCII)
Ø1	Baud Rate: 1 = 1ØØ baud 2 = 15Ø baud 3 = 3ØØ baud 4 = 6ØØ baud 5 = 12ØØ baud 6 = 24ØØ baud 7 = 48ØØ baud 8 = 96ØØ baud

Byte	Contents
	Some applications do not run well at the higher baud rates.
Ø2	Data Word Length (5 - 8 bits)
Ø3	Parity 'E' (even), 'O' (odd), or 'N' (none)
Ø4	Number of stop bits (1 or 2)
Ø5	End of list marker (binary Ø)

Example

```
            .
            .
RS232C      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#RS232C SVC NUMBER
            LDA         .A1,PARAM LIST
            MOVW        @A1,#CHANNEL A
            MOVW        1@A1,#BAUD RATE
            MOVW        2@A1,#DATA WORD LENGTH
            MOVW        3@A1,#EVEN PARITY
            MOVW        4@A1,#STOP BITS
            MOVW        5@A1,#END LIST
            STL         .A1,8@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
RS232C SVC NUMBER
            EQUW        55
PARAM LIST
            RDATAB      6,Ø
CHANNEL A
            EQUB        'A
BAUD RATE
            EQUB        3
DATA WORD LENGTH
            EQUB        7
EVEN PARITY
            EQUB        'E
```

```
STOP BITS
          EQUB      1
END LIST
          EQUB      ØØ
```

SETBRK **Function Code 269**
Set <BREAK>

Lets you enable the <BREAK> key by defining a <BREAK>-key processing program.

Whenever the <BREAK> key is pressed, your processing program takes over. On entry to the <BREAK> processing program, the return address of the interrupted routine is on the User Stack and can be returned to with an RTR instruction (return with restore).

See 'PROGRAMMING WITH USER INTERRUPTS' for further information.

Entry Conditions

Byte-offset	Ø-1	269
	6-7	switch Ø = Off (normal TRSDOS-16 break processing) 1 = On (user programmed break processing) 2 = Disable (TRSDOS-16 ignores <BREAK> key completely)
	8-11	transfer address Ø = Fetch non-zero = Store

Exit Conditions

Byte-offset	2-3	error code

The transfer address is the address you want to jump to when the <BREAK> key is pressed. It must be even.

Example

```
          .
          .
SETBRK    LDØ       .AØ,SVC BLOCK
          MOVW      @AØ,#SETBRK SVC NUMBER
          MOVW      6@AØ,#SWITCH ON
          MOVL      8@AØ,#TRANSFER ADDRESS
```

```
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
              .
              .
SVC BLOCK
            RDATAB      32,Ø
SETBRK SVC NUMBER
            EQUW        269
SWITCH ON
            EQUW        1
TRANSFER ADDRESS
            EQUL        H'8ØØØ
```

SETTRP **Function Code 266**
Sets a Trap Vector

Allows you to set or remove a trap vector. When a BRK is executed (either through the BRK-BRKV instructions or an error trap) control goes to the address specified by the user. See SETBRK and PROGRAMMING WITH USER INTERRUPTS for further information on interrupts.

Entry Conditions

Byte-offset	Ø-1	266
	6-7	function code
	8-9	vector number
	1Ø-13	vector address

Exit Conditions

Byte-offset	2-3	error code
	14-17	previous vector address

The function codes are:
- Ø - Install (SET) trap
- 1 - Remove trap
- 2 - Remove all traps (except BRK Ø)

The available vector numbers are:

Vector #	Assignment
1 - 15	Trap vectors 1-15
16	Illegal Instruction
17	Zero Divide
18	CHK Instruction
19	TRAPV Instruction
2Ø	Privilege Violation
21	Line 1Ø1Ø Emulator
22	Line 1111 Emulator
23	Access out of Partition
24	Odd Address Error

Note that BRK Ø is reserved for implementation of supervisor calls.

The Emulators (Vectors 21 and 22) are illegal opcodes which can be trapped. These are opcodes that have an 'A' or 'F' in the first nybble (high order 4 bits). That is Axxx or Fxxx as an opcode.

The vector address is the address to jump to when the trap is executed. It must be even.

Example

```
            .
            .
SETTRP      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#SETTRP SVC NUMBER
            MOVW        6@AØ,#SET TRAP FUNCTION
            MOVW        8@AØ,#TRAP 16
            MOVL        1Ø@AØ,#TRAP 16 ADDRESS
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
SETTRP SVC NUMBER
            EQUW        266
SET TRAP FUNCTION
            EQUW        Ø
TRAP 16
            EQUW        16
TRAP 16 ADDRESS
            EQUL        H'5ØØØ
```

UNLOCK **Function Code 136**
Unlock Record

Unlocks a specified record number, or all records within a specified file, that you previously locked.

See READNX or DIRRD for information on how to lock a record.

Entry Conditions

Byte-offset	Ø-1	136
	6-7	file identification number
	8-9	record number

Exit Conditions

Byte-offset	2-3	error code

The record number contains the logical record number of the file to be UNLOCKed. Specifying a -1 in the record number byte-offset UNLOCKs all the records in the file which are locked by the calling user.

The file identification number is a unique number assigned to each file when it is opened.

Example

Before executing this program, store the file identification number in register A1.

```
            .
            .
UNLOCK      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#UNLOCK SVC NUMBER
            STW         .A1,6@AØ
            MOVW        8@AØ,#UNLOCK ALL RECORDS
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
            .
```

```
SVC BLOCK
     RDATAB          32,Ø
UNLOCK SVC NUMBER
          EQUW       136
UNLOCK ALL RECORDS
          EQUW       -1
```

VDCHAR **Function Code 8**
Video Character

Outputs a character to the display at the current cursor position.

TRSDOS-16 ignores the control codes not listed in the following chart.

KEY	HEX CODE	FUNCTION
F1	Ø1	Blinking cursor on.
F2	Ø2	Cursor off.
CTRL D	Ø4	Turns on steady cursor.
BACKSPACE	Ø8	Moves cursor back one position and blanks the character at that position.
TAB	Ø9	Advances cursor to next tab position. Tab positions are at 8-character boundaries, 8, 16, 24, 32, ...
CTRL J	ØA	Line feed--cursor moves down to next row, same column position.
CTRL K	ØB	Positions cursor to beginning of previous line.
ENTER	ØD	Moves cursor down to beginning of next line.
CTRL N	ØE	Turns dual routing on.
CTRL O	ØF	Turns dual routing off.
CTRL T	14	Homes cursor (upper left corner)
CTRL W	17	Erases to end of line, cursor doesn't move
CTRL X	18	Erases to end of screen, cursor doesn't move.
CTRL Y	19	Sets Normal Display mode (white on black). Remains Normal until reset by programmer.
CTRL Z	1A	Sets Reverse Display mode (black on white). Remains Reverse until reset by programmer.
ESC	1B	Erases screen and homes cursor (position Ø).
left arrow	1C	Moves cursor back one position.
right arrow	1D	Moves cursor forward one position.
up arrow	1E	Sets 8Ø character line and clears Display.
down arrow	1F	Sets 4Ø character line and clears Display.

Entry Conditions

Byte-offset	Ø-1	8
	6-7	character to output (Ø-127)

Exit Conditions

Byte-offset	2-3	error code

Example

This routine clears the screen.

```
            .
            .
VDCHAR      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#VDCHAR SVC NUMBER
            MOVW        6@AØ,#CLS
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
VDCHAR SVC NUMBER
            EQUW        8
CLS
            EQUW        H'1E
```

VDINIT **Function Code 7**
Video Initialization

You might want to call this routine before starting any I/O to the Video Display. It blanks the screen and homes the cursor (moves the cursor to the upper left corner of the video display).

Entry Conditions

Byte-offset	Ø-1	7
	6-7	character size switch Ø = 4Ø characters/line non-zero = 8Ø characters/line
	8-9	normal/reverse switch Ø = reverse mode (black on green/ black on white) non-zero = normal mode (green on black/white on black)

Exit Conditions

Byte-offset	2-3	error code

Example

```
            .
            .
VDINIT      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#VDINIT SVC NUMBER
            MOVW        6@AØ,#CHAR8Ø
            MOVW        8@AØ,#NORMAL
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
VCINIT SVC NUMBER
            EQUW        7
```

```
CHAR8Ø
                EQUW    1
NORMAL
                EQUW    1
```

VDLINE **Function Code 9**
Video Line

Writes a specified text buffer to the display, starting at the current cursor position. The text buffer must contain codes less than H'8Ø.

Entry Conditions

Byte-offset	Ø-1	9
	6-7	buffer size (Ø - 255)
	8-9	terminator character to be sent after the buffer text
	1Ø-13	buffer address

Exit Conditions

Byte-offset	2-3	error code
	14-15	on error (number of characters not sent)
	16-17	on error (character causing error)

The buffer pointed to by the buffer address should contain ASCII codes in the range Ø to 127.

This routine handles control codes in the buffer in the same manner as in the VDCHAR routine.

Example

This example retrieves the buffer size, terminator character, and buffer Address from the KBLINE SVC routine.

```
            .
            .
          LDW       .A1,12@AØ  *Get length of input line
          LDW       .A2,14@AØ  *Get terminator character
          LDA       .A3,KEYBD BUFFER  *Get address of buffer
VDLINE    LDA       .AØ,SVC BLOCK
          MOVW      @AØ,#VDLINE SVC NUMBER
          STW       .A1,6@AØ
          STW       .A2,8@AØ
          STL       .A3,1Ø@AØ
```

```
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
              .
              .
SVC BLOCK
            RDATAB      32,Ø
VDLINE SVC NUMBER
            EQUW        9
```

VERSION **Function Code 137**
Get Version of Operating System

Determines the version of software currently servicing 68ØØØ requests.

Entry Conditions

Byte-offset	Ø-1	137

Exit Conditions

Byte-offset	2-3	error code
	6-7	major version level
	8-9	minor version level
	1Ø-11	patch level

The major version level is the integral portion of the version number printed during startup.

The minor version level is the fractional portion of this quantity.

The patch level is any level following the version number.

For example, version 2.Øa returns major version level 2, minor version level Ø, and patch level 'a'.

Example

```
            .
            .
VERSION     LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#VERSION SVC NUMBER
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
VERSION SVC NUMBER
            EQUW        137
```

VIDKEY **Function Code 12**
Video Key

This routine combines the functions of VDLINE and KBLINE.

It writes a buffer of data to the display, starting at the current cursor position, then waits for a line from the keyboard.

Once the text message has been displayed, the cursor will be positioned immediately after the last character displayed. To move it to another position, a control code can be placed at the end of the text buffer.

VIDKEY then uses KBLINE to get a line from the keyboard. Note that before starting the line input, all previously stored keystrokes are cleared.

Refer to KBCHAR and VDCHAR for a list of control codes and other details.

Entry Conditions

Byte-offset	Ø-1	12
	6-7	number of characters to be displayed (Ø-255)
	8-9	length of keyboard input field (Ø-255)
	1Ø-13	address of keyboard input buffer
	14-17	address of display text buffer

Exit Conditions

Byte-offset	2-3	error code
	18-19	on error (number of characters not sent)
	2Ø-21	on error (character causing error)

Example

```
            .
            .
VIDKEY      LDA         .AØ,SVC BLOCK
            MOVW        @AØ,#VIDKEY SVC NUMBER
            MOVW        6@AØ,#DISPLAY COUNT
            MOVW        8@AØ,#INPUT COUNT
            LDA         .A1,KEYBD BUFFER
            STL         .A1,1Ø@AØ
            LDA         .A1,MESSAGE PROMPT
            STL         .A1,14@AØ
            BRK         #Ø
            TESTW       2@AØ
            BNE         ERROR
            .
            .
SVC BLOCK
            RDATAB      32,Ø
VIDKEY NUMBER
            EQUW        12
DISPLAY COUNT
            EQUW        21
MESSAGE PROMPT
            TEXT        'ENTER THE ANSWER HERE'
INPUT COUNT
            EQUW        2Ø
KEYBD BUFFER
            RDATAB      2Ø,Ø
```

WRITNX **Function Code 43**
Write Next Record

Writes the next record after the last record accessed; i.e., this routine writes records sequentially.

If WRITNX is the first access after opening the file, it writes the first record.

Entry Conditions

Byte-offset	Ø-1	43
	6-7	file identification number
	8-11	record buffer address

Exit Conditions

Byte-offset	2-3	error code

The file identification number is a unique number assigned to each file when it is opened.

The record buffer address is a 32-bit address that points to the beginning of a 256-byte Record Buffer; that is where the record will be placed after the disk read.

Example

Store the File Identification Number in register A1 before executing this routine.

```
            .
            .
WRITNX      LDA       .AØ,SVC BLOCK
            MOVW      @AØ,#WRITNX SVC NUMBER
            STW       .A1,6@AØ
            LDA       .A2,RECORD BUFFER AREA
            STL       .A2,8@AØ
            BRK       #Ø
            TESTW     2@AØ
            BNE       ERROR
            .
            .
```

```
SVC BLOCK
          RDATAB    32,Ø
WRITNX SVC NUMBER
          EQUW      43
RECORD BUFFER AREA
          RDATAB    256,Ø
```